AD-A259 384

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE

3. REPORT TYPE AND DATES COVERED
FINAL 1 Sep 89 - 31 Jan 92

4. TITLE AND SUBTITLE
"MEMORY-BASED EXPERT SYSTEMS" (U)

5. FUNDING NUMBERS
61102F
2304/A7

6. AUTHOR(S)
Dr. Roger C. Schank

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Northwestern University
Institute for the Learning Sciences
1890 Maple Avenue
Evanston, IL 60201

8. PERFORMING ORGANIZATION REPORT NUMBER
AFOSR-TR- 93 0006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR/NM
Bldg 410
Bolling AFB DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-89-0493

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release;
Distribution unlimited

12b. DISTRIBUTION CODE
UL

13. ABSTRACT (Maximum 200 words)

The goals of this project have been to carry out research aimed at implementing and applying case-based reasoning, or CBR (Riesbeck and Schank, 1989) in a variety of distinct tasks and domains. In particular, the work carried out under this project has focussed on three problems: (1) The development of a robust memory-based parsing technology (Direct Memory Access Parsing, or DMP), (2) The development of case-based systems for creatively approaching complex problems in social and political domains and (3) The application of case-based reasoning in educational settings.

14. SUBJECT TERMS

15. NUMBER OF PAGES
15

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev 2-89)
Prescribed by ANSI Std Z39-18
298-102

Memory-Based Expert Systems:

Final Report

submitted December 1992 to

The Air Force Office of Scientific Research

for

grant no. AFOSR-89-0493

by

Northwestern University
The Institute for the Learning Sciences
1890 Maple Avenue
Evanston, Illinois 60201
(708) 491-3500
fax (708) 491-5258

1 Introduction

The Air Force Office of Scientific Research has funded a project in "Memory-Based Expert Systems" (grant number AFOSR-89-0493) at Northwestern University's Institute for the Learning Sciences, under the direction of Professor Roger C. Schank, Principal Investigator. The goals of the project, as outlined in the original proposal, have been to carry out research aimed at implementing and applying **case-based reasoning**, or **CBR** (Riesbeck and Schank, 1989), in a variety of distinct tasks and domains. In particular, the work carried out under this project has focussed on three problems:

(1) The development of a robust memory-based parsing technology (**Direct Memory Access Parsing**, or **DMAP**).

(2) The development of case-based systems for creatively approaching complex problems in social and political domains.

(3) The application of case-based reasoning in educational settings.

The rest of this document briefly outlines the results of our efforts in these three areas. Details may be found in the accompanying technical reports, and in the references.

2 Direct memory access parsing

Our efforts to transform the theory of memory-based language analysis into a robust, practical technology has led us to develop the **XDMAP** system. XDMAP is a Lisp-based tool for X-windows for building hierarchical dynamic memory structures, such as those described in Schank (1982). Some aspects of the tool are fairly standard: For example, memory structures are displayed as boxes, organized into a family tree, and the user can point and click to inspect structures, add and delete structures, and add and delete links.

Unique to XDMAP, however, is the ability to add parsing information in a similarly simple point and click manner. XDMAP uses the Direct Memory Access Parsing (DMAP) algorithm, as described in Riesbeck and Martin (1985)

and Riesbeck and Schank (1989). In DMAP, you attach templates, such as "**actor** says **info**" and "**actor** told **hearer info**" directly to the knowledge structures that those templates might refer to, such as **communication-event**. The DMAP understanding algorithm's primary job is to read text and find the memory structures referred to by that text. This could mean reading "John gave Mary a kiss" and finding the concept **kiss-event**, but that would be missing the point. A dynamic memory (Schank, 1982) contains episodic knowledge, so reading "John gave Mary a kiss," should lead DMAP to other examples of kissing events, especially those involving John or Mary, if present in memory. More interestingly, reading "Bush orders an end to the fighting," in early 1991 should lead the parser to memory structures describing the Persian Gulf war. This would enable the understanding system to (a) understand what "the fighting" refers to, and (b) update its knowledge of the current state of the world. A standard understanding system would read "Bush orders an end to the fighting," and only recognize that a communication event occurred, and would not be able to tell what kind of activity "the fighting" referred to, because it might be anything from squabbles in Congress to military action, and those are quite different concepts.

The DMAP understanding module in XDMAP provides this ability to find and update episodic knowledge. In addition, the XDMAP tool also provides

- An interface for defining a frame-based hierarchical memory.
- A point-and-click interface for linking text to knowledge units.
- Debugging and maintenance tools.

The general sequence of knowledge entry using XDMAP is as follows:

(1) The user enters a short text (one or more sentences) into the **Input Text** portion of the **Text Info** window.

(2) Pressing the **Parse** button tells XDMAP to parse the text, producing the display in the **Parsed Result** portion of the **Text Info** window, indicating which memory structures were recognized.

(3) If the desired memory structures were not found, the user can point and click to connect segments of the text, e.g., "Persian Gulf," then to the desired memory structure in the **MOP Forest** window.

(4) If the desired memory structure does not exist, the user can add it, using the graphical tools available in the **MOP Forest** window.

(5) Pressing the Parse button again will re-parse the text, showing the new results of the parse, and so on.

XDMAP automatically generalizes the text segments as appropriate to enable to parse texts with similar structure. The user doesn't have to worry about this. Furthermore, the DMAP algorithm automatically handles ambiguous sentences, resolving them according to the memory structures available in memory, so the user doesn't have to worry about that either.

For example, to add "Bush orders an end to the fighting," the user would enter that text, press the **Parse** button, and see, perhaps, that the concept **George-Bush** and the concept **fight-event** were recognized. If the user wants "an end to the fighting" to refer to the concept **cease-fire**, he would select that portion of the text and then select that concept, and tell XDMAP to link them.

Suppose that XDMAP now parses the text to the concept **George-Bush** and the concept **cease-fire**, but not the concept **communication-event**, perhaps because this particular sense of the word "order" is not known. In many systems, the user would have to define "order," specify what kind of verb it is, what kinds of syntactic structures it can appear in, and so on. In XDMAP, the user simply selects the entire sentence, selects the concept B, and links the two. XDMAP then automatically:

(1) Generalizes "Bush orders an end to the fighting," to "**actor** orders **info**," and

(2) Creates an instance of the concept **communication-event**, with the concept **George-Bush** and the concept **cease-fire** filled in the appropriate slots.

The latter action means that the content of this text is in episodic memory, ready for retrieval in future text understandings.

XDMAP has proven to be a robust and relatively easy to use tool for simultaneously building memory structures and developing the parsing knowledge necessary in order to understand natural language texts involving those structures. It is now being employed to develop natural language capabilities for computer-based instructional systems at ILS.

3 Case-based problem-solving in complex domains

Two basic technological issues arise in efforts to apply case-based reasoning to complex problem-solving domains: first, the **indexing** and **retrieval** of complex, highly structured case representations, and second, the creative **adaptation** of such cases to fit the requirements of the new situation. Our efforts to address these issues have primarily been carried out within the context of two innovative CBR systems, **Abby** and **BRAINSTORMER**.

3.1 Abby

Since it constitutes half of the basic CBR process model, progress in CBR requires that we understand how retrieval works in detail. Until we know what we can expect to get from memory, we don't really know how much or what kind of work remains for the rest of the system. The point of CBR is that, ideally, there should be relatively little work left over. Progress in understanding the CBR retrieval step requires that we make headway on the indexing problem.

The development of the **Abby** case-based lovelorn advising system (Domeshek, 1992) has followed that research plan. The result is a program that accepts input describing lovelorn situations, and responds by telling stories of similar situations interpretable (by a human being) as carrying

advice applicable to the input situation. All Abby knows about its cases are their indices: descriptions of conditions under which it makes sense to tell each story. Instead of using a rich case representation to generate novel behaviors specifically tuned to novel circumstances, Abby tells a canned story and leaves it to the user to extract the implicit advice.

To submit a problem to Abby, a user familiar with the system's index formalism must use an editor to create a probe index representing a particular description of the problem. When the user is satisfied that the probe correctly expresses the intended problem, Abby can be asked to search memory for remindings. Abby returns with an ordered list of stories in memory ranked according to how well their labels matched the probe.

Abby currently contains 500 label indices, each giving access to a story rendition; these renditions are ASCII text files telling the tale of some past problem in a way intended to make sense in situations where the attached index has been matched. In all, there are over 250 distinct situations described in these renditions. The stories in Abby's collection were adapted from materials gathered over a period of several years from a variety of sources. The fact that Abby contains hundreds of stories is important for both theoretical and practical reasons. First, the availability of a large number of examples and the need to discriminate amongst them drove the invention of representational vocabulary; the sheer size of the corpus was both a motivation and a test for the indexing system. Second, a large corpus is a prerequisite for a system having any chance at useful performance in a complex domain like lovelorn advising (Schank, 1991b).

An index is defined functionally as a specification of the information that ought to affect a case's retrieval. Unfortunately, using standard composible representations and graph matching techniques, the information accessed through an index could easily remain unbounded. Abby's indices are therefore restricted by limiting substructure, recursion, and information from types, and the explicit encoding of abstract relationships.

A typical example of the sort of problem that Abby can supply advice about is the following: A teenage girl complains *"my parents won't let me marry*

Johnny; they think I'm too young and shouldn't marry yet." Abby's advice for this teenager is the following story:

> *Sometimes parental concern can be overwhelming. I knew a woman of about 30 who finally had to move out of her parent's house because her mother kept pestering her to get married. She liked men well enough and intended to get married some day. She just hadn't met anyone she wanted to marry yet (and who wanted to marry her). She had lots of friends, and she went out on dates fairly often; this was no social misfit or recluse. But whenever she got home, her mother was waiting up for her to ask if she had met anyone who interested her. It finally just got to be too much for her, so she had to find a place of her own. While that was probably an overdue move for her anyway, the mother could have avoided forcing the issue if she wanted to keep her daughter around.*

Addressed to the girl, the simplest interpretation of this story is that if the teenager doesn't like her parents' interference in her love-life then she should move out. Notice that the mismatch between the ages of the teenage girl and the 30-year-old may make this direct advice inapplicable. Yet that very mismatch and inapplicability can lead the girl to a more useful conclusion: If she is not old enough to move out on her own, then she may not be old enough to ignore her parents' wishes.

By packaging task-relevant features into a fixed index frame, Abby addresses the constraints of the indexing problem—recover **relevant** cases **quickly** from a **large** memory—plus the requirement for an explicit theory of index content in the complex social domain where relevance often depends on abstract intentional features. The result is a comprehensible and usable index formalism. The simple exercise of coding more than 500 indices using Abby's index formalism and representational vocabulary has served as a test for the theory that is, in many ways, a more rigorous test of expressivity than that to which most representational systems are ever subjected. In fact the last 250 indices were coded with little change to the index format: The system has, it seems, converged on an adequate index design for complex social problem-solving tasks.

3.2 BRAINSTORMER

To study the computational underpinnings of case adaptation, we have constructed the **BRAINSTORMER** system (Jones, 1992). BRAINSTORMER is a planner that operates in the domain of terrorist crisis management. The system is handed a planning problem and a large set of proverbs, and uses the proverbs to help it plan. BRAINSTORMER first tries to solve problems it is given on its own, then considers its proverbs one by one in light of whatever difficulties it has encountered. The idea is to generate lots of suggestions without too much concern for their plausibility, with the hope that something interesting will turn up. BRAINSTORMER thus implements a theory of **brainstorming**, in the ordinary sense of the term: a technique for solving hard problems by generating lots of ideas.

While BRAINSTORMER knows something about terrorism, it is not an expert in this domain. Instead, BRAINSTORMER'S expertise lies in the more general domain of planning and social interaction. In this sense, all problems having to do with terrorism appear novel to BRAINSTORMER. Just as people fall back on general knowledge to solve novel problems, BRAINSTORMER adapts its abstract knowledge about planning and acting to suggest ways to prevent terrorist attacks.

One way to think of BRAINSTORMER is as a **creativity aid**: a system that can assist a user presented with a difficult problem by suggesting large numbers of possible solutions. The system is equipped with a memory of culturally-shared cases, and generates large numbers of plausible solutions to problems by the simple strategy of trying to adapt each case to fit. From the user's point of view, then, BRAINSTORMER takes a planning problem as input and produces a large number of sketchy solutions, some of which the user might not have thought of on his own. Of course, the solutions that BRAINSTORMER actually generates are a function of its memory of culturally-shared cases, even though these are hidden from the user. It follows that from a programming standpoint, it is more useful to think of BRAINSTORMER's input as a user-supplied planning problem and a set of proverbs, and its output as a set of planning suggestions produced with the help of these proverbs.

Suppose, for example, a user hands BRAINSTORMER the planning problem paraphrased as follows:

Situation: A terrorist attack carried out by terrorists from the P.L.O., in which a group of people was held hostage, and one of the hostages was killed.

Goals: Prevent recurrence of future terrorist attacks with similar causes and problematic effects; retaliate against agents causally implicated in producing these effects.

Based on representations of the following proverbs, BRAINSTORMER will generate the following corresponding plans, among others:

Proverb: Covetousness is at the root of all evil.
Plan: Help the terrorists achieve their goal of a Palestinian state.
Plan: Persuade the Palestinians that they don't need a Palestinian state.

Proverb: When in Rome, do as the Romans do.
Plan: Adopt the methods of the terrorists.

Proverb: He that has suffered more than is fitting will do more than is lawful.
Plan: Improve living conditions in the refugee camps.
Plan: Get the Palestinians out of the refugee camps.

Proverb: An eye for an eye, a tooth for a tooth.
Plan: Kill or arrest the terrorists.

Proverb: No trouble but a priest is at the bottom of it.
Plan: Assassinate or arrest Khomeini.

Proverb: An old poacher makes the best keeper.
Plan: Employ a former terrorist as an advisor on countering terrorism.

Proverb: Cities are taken by the ears *(refers to the value of propaganda).*
Plan: Censor news reports of the terrorists' activities.

Proverb: Fish begin to stink at the head.
Plan: Destroy the political power of the P.L.O. leaders.

Proverb: Anger punishes itself.
Plan: Do nothing (the terrorists will self-destruct).

Proverb: Danger makes men devout.
Plan: Get the terrorists' religious leaders to denounce terrorism.

Some of these plans may seem a little implausible: The Palestinians, for example, are in reality unlikely to abandon their goal of obtaining a Palestinian State. A number of the others, however are interestingly perceptive or even creative. It is left to the user to filter out the promising candidates from the unpromising ones. Brainstorming is a kind of generate and test approach to creative problem solving; BRAINSTORMER assists the "generate" part of this process. The system is not intended to produce complete, detailed solutions to problems: Instead, it is designed to suggest promising avenues for the user to explore.

Conceptually, plan generation in BRAINSTORMER divides into three stages:

(1) **Initial planning:** BRAINSTORMER's planning component attempts to generate appropriate plans on its own, without the help of any proverbs. When BRAINSTORMER can make no further progress, however, it turn to proverbs for assistance.

(2) **Adaptation:** A proverb is adapted to help the planner in some particular way, by providing it with a memory structure that allows it to make further progress.

(3) **Final planning:** The planner carries out planning steps enabled by adaptation, eventually producing one or more concrete plan suggestions.

This process is iterative: More than one proverb can contribute to generating a single plan.

On several occasions, BRAINSTORMER's adapter spontaneously produced suggestions that we found surprising, using general adaptation capabilities that we had constructed earlier to handle other examples. For example, when adapting the proverb *he who has suffered more than is fitting will do more than is lawful,* BRAINSTORMER suggested an explanation that we had not anticipated. After hypothesizing that suffering led the terrorists to carry out the terrorist attack, the system adapted the same proverb a second time, this time generating a motivational explanation for the suffering: Perhaps some other agent deliberately caused the Palestinians to suffer, with the aim of getting them to carry out illegal actions like the terrorist attack. We did not expect BRAINSTORMER to come up with this explanation, but when it did, we asked ourselves whether it is plausible. In fact, it is: It can be argued that the Palestinians are languishing in the refugee camps because of a concerted policy of neighboring Arab states. These countries have strictly limited immigration of Palestinian refugees, in part because they hope that the refugees will blame Israel for their plight, and therefore harass Israel by every means at their disposal (including terrorism). On this view, the Palestinian refugees are little more than pawns in an ongoing regional power play initiated by unscrupulous Arab states.

Although as outside observers we have sufficient background knowledge about Middle Eastern politics to elaborate the adapter's sketchy explanation, BRAINSTORMER does not do much with it, because it lacks the relevant domain-specific knowledge. As a consequence, BRAINSTORMER cannot go on to generate plans that might follow from the elaborated explanation, such as *convince the Palestinians that Israel is not the source of their difficulties.* Nevertheless, BRAINSTORMER'S unexpected proposal got us to think about the terrorist problem in a new way, which, of course, is a primary aim of a creativity aid.

BRAINSTORMER is only a prototype of a practical creativity aid: It is equipped with a case library of 15 culturally-shared cases. It does however

illustrate the essential components of a larger system. In particular, it is capable of flexibly adapting each of its proverbs to produce a number of different kinds of outputs, each of which can potentially assist a planner in a different way.

4 Case-based teaching

Experts are repositories of cases. Becoming an expert thus depends upon acquiring this case base in a usable form. **Case-based teaching** is based upon the idea that good teaching is good story telling (Schank, 1990): Students respond well to stories if they are told in an interesting way and at an appropriate time. The case-based teaching architecture exploits the basic capacity for students to learn from stories and the basic desire of teachers to tell stories that illustrate their experiences.

There are two major issues in this approach to teaching: First is the construction and exploitation of a case base from which cases can be drawn when they are needed. The second is the creation of a situation that would cause a student to be interested in hearing about a relevant case. The case base must be indexed in such a way as to relate to the situations that are created for the student.

The general paradigm we have developed for case-based teaching is as follows:

(1) **Situation:** Present an interesting situation for a student to try out.

(2) **Failure:** Allow him to fail.

(3) **Indexing:** Have a story ready to tell that relates to what the student did wrong.

(4) **Story telling:** Tell story when consulted.

We have built a number of case-based teaching systems along these lines, in a wide variety of domains, and for a diverse set of users (Schank, 1991a). One of

these **CREANIMATE**, attempts to teach simple functional anatomy to elementary school students (Edelson, 1991). The program asks the student to design a new animal. The animal that the student proposes forms the basis for a dialog about the physical features of animals and how those features help them to survive in the wild. Thus, for example, if a student wants to create a kangaroo that flies, the program would initiate a discussion about why animals fly, that would be illustrated by professionally produced video clips illustrating different animals that use flying in different ways. After the student decides on a reason why he wants his kangaroo to fly, he could continue to discuss the physical changes that would be required in order for a kangaroo to fly, the particular type of flying that it might be useful for a kangaroo to be able to carry out, and how flying would change the things that real kangaroos currently do. During the discussion, general principles would always be illustrated by video clips of particular animals.

CREANIMATE instantiates the above paradigm for case-based teaching as follows:

(1) **Situation:** The situation consists of a design for a new animal proposed by the student. This is a particularly effective educational situation because it is generated entirely by the student and provides him with a personal investment in the outcome. The student becomes concerned with the viability of his own creation. Children will readily tell you an animal that they would like to design if they could: The trick here is simply to bring to the fore the kind of goal that children will readily work towards.

(2) **Failure:** The program and the student conduct a dialog in order to explore the ramifications of the student's design. This discussion may uncover actual failures in the design. For example, simply adding wings to a cow is not sufficient to enable it to fly. More importantly, the process of seeking failures, even if there are none in the design, leads the student through an inquiry process that will expose him to important principles and stories.

(3) **Indexing:** The indexing scheme for the domain must be consistent with the knowledge representation of the domain employed by the dialog manager. The stories are indexed in two ways: first, according to the specific

animals that appear in the video clip using features that achieve their behaviors, and second, according to the general principles they illustrate. Thus the program is able to look at a proposed design for an animal, identify which general principles apply to that design, search the case base for stories that illustrate those general principles, and then make the connection between the specifics of the video story and the specifics of the student's proposal, in order to make the video relevant for the student.

(4) **Story telling:** The issue for this program is not so much story telling as story selection. The stories exist in video archives already (we do not need to go out into the wild to film animals). However, we do need to carefully select the stories that go into the program, using the following criteria:

- **Excitment**—The program succeeds or fails on the vividness of its video. Fortunately, it is not hard to find exciting video footage of animals.

- **Instructiveness**—The video must present its lesson clearly and effectively.

- **Coverage**—We must have a large and diverse enough library of video to cover the multitude of ideas that students generate, and a sufficient number of the actual features and behaviors that exist in nature.

Our key insight in constructing this system has been that a teacher, whether human or machine, can never have a complete collection of stories for any domain. With a limited number of stories, it is impossible to have the perfect story for any situation. Thus, it is important to be able to find a story that is similar enough to be appropriate, if imperfect. The way to ensure that you can find these appropriate stories is to have indices that are written in a scheme that is expressive enough so that, in the absence of a perfect story for a situation, indices can be considered in part in order to find stories that are relevant.

5 References

Domeshek, E. 1992. Do the right thing: A component theory for indexing stories as social advice. Technical report no. 26, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Edelson, D. 1991. Why do cheetahs run fast? Responsive questioning in a case-based teaching system. In *Proceedings of the 1991 International Conference on the Learning Sciences,* Association for the Advancement of Computing in Education, Charlottesville, VA, pp. 138-144.

Jones, E. 1992. The flexible use of abstract knowledge in planning. Technical report no. 28, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Riesbeck, C., and Martin, C. 1985. Direct memory access parsing. Research report no. 354, Yale University, Dept. of Computer Science, New Haven, CT.

Riesbeck, C., and Schank, R., eds. 1989. *Inside Case-Based Reasoning.* Lawrence Erlbaum Associates, Hillsdale, NJ.

Schank, R. 1982. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People.* Cambridge University Press, Cambridge, England.

Schank, R. 1990. *Tell Me a Story: A New Look at Real and Artificial Memory.* Charles Scribner's Sons, New York.

Schank, R. 1991a. Case-based teaching: Four experiences in educational software design. Technical report no. 7, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Schank, R. 1991b. Where's the AI? Technical report no. 16, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.